Statistical Research Skill

Group 6

School of Mathematics, University of Edinburgh

Model Structure

The Bayesian hierarchical spatio-temporal (BHST) model is specified as:

$$y_{s,t} = \mu + u_s + \delta(t) + \gamma_t + \epsilon_{s,t},$$

where:

- $y_{s,t}$ denotes the observed temperature at spatial location s and time t.

Model Components

- **Global mean (μ)**
  Represents the overall average temperature level across all time periods and spatial locations.
- **Smooth temporal trend ($\delta(t)$)**
  Captures long-term climate evolution over time.
  A second-order random walk (RW2) prior is used to allow a flexible, smooth nonlinear trend without imposing a strict linear structure.
- **Spatial structured effect (u_s)**
  Models spatial dependence between neighbouring grid cells.
  This component is specified using a Conditional Autoregressive (CAR) or BYM2 prior, allowing geographically nearby locations to exhibit similar temperature patterns.
- **Temporal autocorrelation (γ_t)**
  Accounts for short-term persistence between consecutive observations.
  This component is modelled using an autoregressive process of order one (AR(1)).
- **Observation error ($\epsilon_{s,t}$)**
  Represents unexplained variability, assumed to follow:

$$\epsilon_{s,t} \sim \mathcal{N}(0, \sigma^2).$$

The model decomposes temperature variability into: long-term temporal trend, spatial structure, short-term temporal dependence (?), and residual noise (?).

```
# ============================================================
# Scotland monthly grid anomalies (NetCDF) -> INLA models
# - Handles ocean/outside-domain NaNs properly
# - Uses anomaly as response (recommended)
# - Train: 1901-2020, Test: 2021-2024 (held out as NA for prediction)
# - Models:
#   M0: baseline
```

```
#   M1: baseline + AR1 short-term + RW2 trend
#   M2: baseline + AR1 short-term + RW2 trend + BYM2 spatial
# ============================================================

library(terra)
```

R Code

```
## Warning: package 'terra' was built under R version 4.5.2

## terra 1.8.93
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.5.2

##
## Attaching package: 'data.table'

## The following object is masked from 'package:terra':
##
##     shift
```

```
library(spdep)
```

```
## Warning: package 'spdep' was built under R version 4.5.2

## Loading required package: spData

## Warning: package 'spData' was built under R version 4.5.2

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`

## Loading required package: sf

## Warning: package 'sf' was built under R version 4.5.2

## Linking to GEOS 3.13.1, GDAL 3.11.4, PROJ 9.7.0; sf_use_s2() is TRUE
```

```
library(INLA)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.5.2

## This is INLA_25.10.19 built 2025-10-19 19:10:20 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - List available models/likelihoods/etc with inla.list.models()
##  - Use inla.doc(<NAME>) to access documentation
```

```
# ----------------------------
# A) Read NetCDF + build long table (cell x month)
# ----------------------------
r_anom <- rast("ds_scotland.nc", subds = "anom")
r_tmp  <- rast("ds_scotland.nc", subds = "tmp")

tt <- time(r_anom)
if (is.null(tt)) {
  tt <- seq(as.Date("1901-01-01"), by = "month", length.out = nlyr(r_anom))
}
```

```r
time(r_anom) <- tt
time(r_tmp)  <- tt

df_anom <- as.data.frame(r_anom, xy = TRUE, cells = TRUE, na.rm = FALSE)
df_tmp  <- as.data.frame(r_tmp,  xy = TRUE, cells = TRUE, na.rm = FALSE)

layer_dates <- format(tt, "%Y-%m")
setnames(df_anom, old = names(df_anom)[-(1:3)], new = layer_dates)
setnames(df_tmp,  old = names(df_tmp)[-(1:3)],  new = layer_dates)

dtA <- melt(as.data.table(df_anom),
            id.vars = c("cell","x","y"),
            variable.name = "ym", value.name = "anomaly")

dtT <- melt(as.data.table(df_tmp),
            id.vars = c("cell","x","y"),
            variable.name = "ym", value.name = "tmp")

df <- merge(dtA, dtT, by = c("cell","x","y","ym"), all = TRUE)
setnames(df, old = c("x","y"), new = c("lon","lat"))

df[, year  := as.integer(substr(ym, 1, 4))]
df[, month := as.integer(substr(ym, 6, 7))]

# stable time index (use sorted unique ym)
ym_levels <- sort(unique(df$ym))
df[, t_id := match(ym, ym_levels)]

df[, y_obs := anomaly]  # response variable

# ----------------------------
# A2) Domain filtering: remove cells (NA for ALL months)
# ----------------------------
cell_ok <- df[, .(keep = any(!is.na(y_obs))), by = cell][keep == TRUE, cell]
df <- df[cell %in% cell_ok]

# Re-index spatial id AFTER filtering
df[, area_id := as.integer(factor(cell))]

# ----------------------------
# B) Train/test split + create df_inla for prediction
# ----------------------------
df[, set := fifelse(year <= 2020, "train", "test")]

df_truth <- copy(df)  # keep the true observations for scoring

df_inla <- copy(df)
df_inla[set == "test", y_obs := NA_real_]  # hold out

S <- max(df_inla$area_id)

# space-time iid index (kept in case you use it later)
df_inla[, st_iid := (t_id - 1L) * S + area_id]
```

```r
# ---- NEW: AR(1) index (replicated by area) ----
# Use a separate name to avoid confusion with RW2 index
df_inla[, t_ar := t_id]

# ----------------------------
# C) Build adjacency graph for filtered grid
#    Make rook adjacency on full lattice, then map to area_id and drop NA cells.
# ----------------------------
cells <- unique(df_inla[, .(area_id, lon, lat)])
setorder(cells, lat, lon)

ux <- sort(unique(cells$lon))
uy <- sort(unique(cells$lat))
ncol <- length(ux)
nrow <- length(uy)

cells[, col := match(lon, ux)]
cells[, row := match(lat, uy)]

mat <- matrix(NA_integer_, nrow = nrow, ncol = ncol)
mat[cbind(cells$row, cells$col)] <- cells$area_id

# lattice index -> area_id (or NA)
idx_to_area <- as.vector(mat)

# rook adjacency on the FULL lattice (indices 1..(nrow*ncol))
nb_full <- cell2nb(nrow, ncol, type = "rook")

# Build nb list in AREA-ID space (1..S)
nb_area <- vector("list", length = S)

for (idx in seq_along(nb_full)) {
  a <- idx_to_area[idx]
  if (is.na(a)) next

  neigh_idx <- nb_full[[idx]]
  neigh_area <- idx_to_area[neigh_idx]
  neigh_area <- neigh_area[!is.na(neigh_area)]
  neigh_area <- sort(unique(as.integer(neigh_area)))

  nb_area[[a]] <- neigh_area
}

# ensure no NULL entries (INLA graph writer likes integer(0))
for (i in seq_len(S)) if (is.null(nb_area[[i]])) nb_area[[i]] <- integer(0)

write_inla_graph <- function(nb, file) {
  S <- length(nb)
  con <- file(file, open = "wt")
  on.exit(close(con), add = TRUE)

  writeLines(as.character(S), con)
  for (i in seq_len(S)) {
```

```
    neigh <- nb[[i]]
    if (is.null(neigh)) neigh <- integer(0)
    neigh <- sort(unique(as.integer(neigh)))
    line <- paste(c(i, length(neigh), neigh), collapse = " ")
    writeLines(line, con)
  }
}

graph_file <- file.path(tempdir(), "scotland_grid.graph")
write_inla_graph(nb_area, graph_file)
g <- inla.read.graph(graph_file)

# ----------------------------
# D) Fit models (Gaussian anomalies)
# IMPORTANT: fit on df_inla (train+test with NA) so fitted values align for RMSE
# ----------------------------
ctrl_pred <- list(compute = TRUE, link = 1)
ctrl_comp <- list(dic = TRUE, waic = TRUE, cpo = FALSE)
ctrl_fxd  <- list(mean = list("(Intercept)" = 0),
                  prec = list("(Intercept)" = 1/100))  # var=100

# M0: baseline
m0 <- inla(
  y_obs ~ 1,
  data = df_inla, family = "gaussian",
  control.fixed = ctrl_fxd,
  control.predictor = ctrl_pred,
  control.compute = ctrl_comp
)

# M1: baseline + AR1 short-term + RW2 trend
# - RW2 gives a smooth long-term temporal trend shared across space
# - AR1 (replicate=area_id) gives cell-specific short-term persistence
m1 <- inla(
  y_obs ~ 1 +
    f(t_id, model = "rw2", constr = TRUE) +
    f(t_ar, model = "ar1", replicate = area_id),
  data = df_inla, family = "gaussian",
  control.fixed = ctrl_fxd,
  control.predictor = ctrl_pred,
  control.compute = ctrl_comp
)

# M2: baseline + AR1 short-term + RW2 trend + BYM2 spatial
m2 <- inla(
  y_obs ~ 1 +
    f(t_id, model = "rw2", constr = TRUE) +
    f(t_ar, model = "ar1", replicate = area_id) +
    f(area_id, model = "bym2", graph = g, scale.model = TRUE, constr = TRUE),
  data = df_inla, family = "gaussian",
  control.fixed = ctrl_fxd,
  control.predictor = ctrl_pred,
  control.compute = ctrl_comp
```

```
)

# ALL INTERACTION MODELS ARE STILL ERROR----------------------------------------
# M3a - time varying spatial field
#df_inla[, area_tv := area_id]

#m3_tvsp_ar1 <- inla(
#  y_obs ~ 1 +
#    f(t_id, model="rw2") +
#    f(area_tv, model="besag", graph=g, group = t_id,
#      control.group = list(model="ar1")),
#  data=df_inla, family="gaussian",
#  control.fixed=ctrl_fxd,
#  control.predictor=ctrl_pred,
#  control.compute=ctrl_comp,
#)

# M3b space X time interaction with random slope per area (space-specific trend)
# df_inla[, t_center := t_id - mean(t_id)]
# df_inla[, area_slope := area_id]

#m3_slope <- inla(
#  y_obs ~ 1 +
#    f(t_id, model="rw2") +
#    f(area_id, model="bym2", graph=g, scale.model=TRUE) +
#    f(area_id, t_center, model="iid"),  # random slope
#  data=df_inla, family="gaussian",
#  control.fixed=ctrl_fxd,
#  control.predictor=ctrl_pred,
#  control.compute=ctrl_comp,
#)

# ---- Time-varying spatial interaction (SUPPORTED approach)
# Use a structured interaction: Besag evolving in time (group RW1 / AR1)
# (bym2 + group is often not supported; besag + group is)
#df_inla[, area_tv := area_id]

# explicit index for clarity
#m3_tvsp_rw1 <- inla(y_obs ~ 1 +
#                      f(t_id, model = "rw2") +
#                      f(area_id, model = "bym2", graph = g, scale.model = TRUE) +
#                      f(area_tv, model = "besag", graph = g, group = t_id,
#                        control.group = list(model = "rw1")),
#                    data = df_inla, family = "gaussian",
#                    control.fixed = ctrl_fxd,
#                    control.predictor = ctrl_pred,
#                    control.compute = ctrl_comp)

#m3_tvsp_ar1 <- inla(y_obs ~ 1 +
#                      f(t_id, model = "rw2") +
#                      f(area_id, model = "bym2", graph = g, scale.model = TRUE) +
#                      f(area_tv, model = "besag", graph = g, group = t_id,
#                        control.group = list(model = "ar1")),
```

```
#                     data = df_inla, family = "gaussian",
#                     control.fixed = ctrl_fxd,
#                     control.predictor = ctrl_pred,
#                     control.compute = ctrl_comp)

# ----------------------------
# E) Compare WAIC/DIC + Test RMSE (2021-2024)
# ----------------------------
get_metrics <- function(mod){
  data.table(
    DIC  = mod$dic$dic,
    WAIC = mod$waic$waic
  )
}

# Generic scorer for test set
score_test <- function(mod, df_inla, df_truth){
  idx_test <- which(df_inla$set == "test")

  # For Gaussian + identity link:
  pred  <- mod$summary.linear.predictor$mean[idx_test]
  truth <- df_truth$y_obs[idx_test]

  ok <- !is.na(truth) & !is.na(pred)
  pred <- pred[ok]; truth <- truth[ok]

  data.table(
    n_test = length(truth),
    RMSE = sqrt(mean((pred - truth)^2)),
    MAE  = mean(abs(pred - truth))
  )
}

mods <- list(M0 = m0, M1 = m1, M2 = m2)

metrics <- rbindlist(lapply(names(mods), function(nm){
  cbind(model = nm, get_metrics(mods[[nm]]))
}))

test_scores <- rbindlist(lapply(names(mods), function(nm){
  cbind(model = nm, score_test(mods[[nm]], df_inla, df_truth))
}))

print(metrics[order(WAIC)])
```

```
##     model       DIC      WAIC
##    <char>     <num>     <num>
## 1:     M2 -54847.70 -54745.84
## 2:     M1 -54742.02 -54647.21
## 3:     M0 192435.65 192435.81
```

```
print(test_scores[order(RMSE)])
```

```
##     model n_test     RMSE      MAE
```

```
##    <char>  <int>    <num>    <num>
## 1:     M0   2112 1.365778 1.213835
## 2:     M2   2112 2.981840 2.410872
## 3:     M1   2112 2.990228 2.417237
```

```
# ----------------------------
# Prefer the model that:
# 1) has lower WAIC (and DIC broadly agrees),
# 2) has lowest Test RMSE,
# 3) is stable (no weird hyperparameters; e.g., AR1 rho not stuck at ~1 with huge variance)
#
# Usually:
# - If M3_tvsp_ar1 improves RMSE and WAIC: pick it (persistent spatial evolution).
# - If AR1 becomes unstable or barely improves: pick M3_tvsp_rw1 (smoother, often more stable).
# - If neither improves much: keep M2 (interpretable + parsimonious).

# ---------- Plot posterior mean components ----------
```